Trigger ve Stored Procedure Uygulamaları

Not
vtys2_h06_2010-11.sql isimli dosyayı sql server'da çalıştırınız.

Örnek 1
EXISTS Fonksiyonu uygulaması

```
DECLARE @SipNo int
SELECT @SipNo=1
IF EXISTS(select * from Satis where
SipNo=@SipNo)
	PRINT 'Kayıt var'
ELSE
	PRINT 'Kayıt Yok'

-- @SipNo değişkeninin değerini 10
yapınız
```

Örnek 2
NOT EXISTS Fonksiyonu uygulaması

```
DECLARE @SipNo int
SELECT @SipNo=1
IF NOT EXISTS(select * from Satis where
SipNo=@SipNo)
	-- Değer True
	PRINT 'Kayıt yok'
ELSE
	-- Değer False
	PRINT 'Kayıt Var'

-- @SipNo değişkeninin değerini 10
yapınız
```

Örnek 3
Satış tablosuna yeni veri girildiğinde çalışacak ve veri eklendi mesajı verecek bir trigger yazınız.

```
CREATE TRIGGER trgSatisGir
ON Satis
AFTER INSERT
AS
BEGIN
PRINT 'Bu Mesaj Trigger dan Gelmektedir.
	Satýþ Tablosuna Yeni Veri Eklendi'
END
```

Örnek 4
Örnek 3 deki trigger'ın çalışmasını sağlamak için Satis tablosuna veri giriniz.

```
INSERT Satis (SipNo, UrunNo,
Miktar,GirisCikis) values
(10,'E',1,0)
```

Örnek 5
Ornek 3 deki triggerda, girilen verideki değeri bir değişkene aktarıp bu değişkenin değerini yazdırıcak şekilde yeniden düzenleyiniz.

```
ALTER TRIGGER trgSatisGir
ON Satis
AFTER INSERT
AS
BEGIN
	-- Değişkenleri tanımlayalım
DECLARE @SipNo int, @UrunNo char(4),
@Miktar float, @GirisCikis tinyint

-- Girilen kayıttaki bilgileri değişkene
aktaralım
SELECT @SipNo=SipNo, @UrunNo=UrunNo,
@Miktar =Miktar, @GirisCikis =GirisCikis
FROM inserted

-- Değişkenleri yazdıralım
PRINT 'SipNo='+ltrim(str(@SipNo)) +
', UrunNo='+ @UrunNo +
', Miktar='+ltrim(str(@Miktar))
END

-- str fonk,  sayıyı metine dönüştürür
-- ltrim : metinin solundaki boşlukları
siler
```

Örnek 6
Örnek 5 deki trigger'ın çalışmasını sağlamak için bir kaç tane farklı Satis tablosuna veri giriniz.

```
INSERT Satis (SipNo, UrunNo,
Miktar,GirisCikis) values
(270,'F',1,0)
```

Örnek 7
Satis tablosuna veri girildiğinde çalışan bir trigger yazınız. Bu trigger girilen urun, ürün tablosunda yok ise kaydı iptal etsin.

```
ALTER TRIGGER trgSatisGir
ON Satis
AFTER insert
AS
BEGIN
	-- Deðiþken tanýmý
	DECLARE @UrunNo varchar(10)
```

```
-- Girilen ürünü değişkene atayalım
SELECT @UrunNo=UrunNo from inserted

-- değişken içindeki ürün yok ise
IF NOT EXISTS (select * from Urun where
UrunNo=@UrunNo)
BEGIN
      PRINT @UrunNo+' kodlu ürün, Urun
Tablosunda Yok. İşlem iptal edildi'
      ROLLBACK TRAN
END
END
```

Örnek 8

Örnek 7 deki trigger'ın çalışmasını sağlamak için, bir kaç tane farklı Satis tablosuna veri giriniz.

```
INSERT Satis (SipNo, UrunNo,
Miktar,GirisCikis) values
(30,'G',11,0)
go
INSERT Satis (SipNo, UrunNo,
Miktar,GirisCikis) values
(30,'A',11,0)
Go
```

Örnek 9

Satis tablosuna veri girildiğinde çalışacak bir trigger yazınız. Bu trigger girilen ürün ürün tablosunda yok ise, ürün tablosuna eklecektir. Var ise GirisCikis deðerine bağlı olarak ürün miktarını arttırıp, azaltsın. GirisCikis deðeri 0 ise artsın, 1 ise azalsın.

```
ALTER TRIGGER trgSatisGir
ON Satis
AFTER insert
As
BEGIN
-- Deðiþkenler
DECLARE @UrunNo varchar(10), @miktar
float , @GirisCikis tinyint

-- Değişkenlere bilgiyi aktaralım
Select @UrunNo=UrunNo, @miktar=miktar,
@GirisCikis=GirisCikis from inserted

-- Ürün varmı?
IF exists(select * from urun where
urunno=@UrunNo)
Begin
-- var
IF @GirisCikis=0
Update urun set StokMiktar=StokMiktar+
@miktar where UrunNo=@UrunNo
ELSE
Update urun set StokMiktar = Stokmiktar
- @miktar where UrunNo=@UrunNo
End
ELSE  --  Urun Yok
INSERT URUN(UrunNo, StokMiktar) values
(@UrunNo, @miktar)
END
```

Örnek 10

Örnek 9 deki trigger'ın çalışmasını sağlamak için bir kaç tane farklı Satis tablosuna veri giriniz.

```
INSERT Satis (SipNo, UrunNo,
Miktar,GirisCikis) values
(30,'G',11,0)
go
INSERT Satis (SipNo, UrunNo,
Miktar,GirisCikis) values
(30,'A',11,0)
go

-- Ürün tablosunu listeleyerek.
inceleyiniz.
SELECT * FROM Urun
```

Örnek 11

Satis tablosundeki ver değiştiğinde çalışacak bir trigger yazınız. Önceki miktar ile sonraki miktarın farkını Urun tablosuna yazsın.

```
USE OrnekLtd
GO
CREATE TRIGGER trgSiparisDegistir
ON Satis FOR UPDATE AS
-- Gerekli değişkenler
DECLARE @UrunNo varchar(10),
@OncekiMiktar float
DECLARE @SonrakiMiktar float

-- Kayıt değişmeden önceki bilgiler
SELECT @UrunNo=UrunNo,
@OncekiMiktar=Miktar FROM deleted
-- Kayıt değiştikten sonraki bilgiler.
SELECT @SonrakiMiktar=Miktar FROM
inserted

-- Aradaki farkı Stok miktarına ekleyen
-- kod.
UPDATE Urun SET StokMiktar=StokMiktar +
        @OncekiMiktar-@SonrakiMiktar
WHERE UrunNo=@UrunNo

Go
```

Trigger ve Stored Procedure Uygulamaları

Örnek 12

```
Örnek 11'daki trigger'ın tetiklenmesi
için aşağıdaki veriyi giriniz.

-- Tabloların ilk durumu
SELECT * FROM Satis  WHERE SipNo=3
SELECT * FROM Urun
-- Veri girişi
UPDATE Satis SET Miktar=30 WHERE SipNo=3
--Girildikten sonraki durumu
SELECT * FROM Urun
```

Örnek 13

Fatura tablosunda veri değiştiren bir stored procedure yazınız. ÜrünNo ve mno' su bilinen kayıtlarda miktar ve fiyatını değiştiren ve kaç kayıtın değiştiği bilgisi parametreden geri dönsün.

```
CREATE PROCEDURE spDegistir
-- Giriş parametreleri
      @mno int,
      @UrunNo varchar(10),
      @Miktar float,
      @Fiyat float,
-- Geridönüþ parametresi
      @KayitSayisi int OUTPUT
AS
UPDATE Fatura SET miktar=@Miktar,
fiyat=@Fiyat WHERE mno=@mno and
UrunNo=@UrunNo

-- Update edilen kayıt sayısı
SELECT @KayitSayisi=@@ROWCOUNT
GO
```

Örnek 14

Örnek 13 deki prosedürü çalıştırınız.

```
DECLARE @Sayi int

-- @Sayı prosedürden geri dönen değer
EXEC spDegistir 1,'A',10,1,@Sayi OUTPUT

-- Geri dönen değerin yazılması
SELECT @Sayi

SELECT * FROM Fatura
```

Örnek 15

-- Örnek 13 deki prosedürü aşağıdaki gibi çalıştırınız.

```
DECLARE @Sayi int

-- @Sayý prosedürden geri dönen deðer
EXEC spDegistir
@mno=3,@UrunNo='B',@Miktar=10,@Fiyat=1,@
KayitSayisi=@Sayi OUTPUT

-- Geri dönen deðerin yazýlmasý
SELECT @Sayi

SELECT * FROM Fatura
```

Çalıştırırken parametre adı yazılırsa istenilen sırada olabilir.

Örnek 13 deki prosedürü aşağıdaki gibi Çalıştırınız.

```
DECLARE @Sayi int

-- @Sayı prosedürden geri dönen değer
EXEC spDegistir @UrunNo='E', @mno=4,
@Fiyat=100, @Miktar=900,
@KayitSayisi=@Sayi OUTPUT

-- Geri dönen değerin yazılması
SELECT @Sayi
Go
-- Fatura tablosu içeriği
SELECT * FROM Fatura
```

Örnek 16

A grubu müşterilerin mno değerini bulunuz ve bulduğunuz mno değeri olan müşterilerin miktar ve tutar toplamını veren sorguyu yazınız.

```
-- Grubu A olanları veren sogu
SELECT mno FROM musteri WHERE Grubu='A'
go

-- İstenen 2. sorgu
SELECT mno,SUM(miktar) as MiktarToplam,
      SUM(miktar*fiyat) as TutarToplam
      FROM Fatura WHERE mno in (1,3)
GROUP BY mno
go
```

Trigger ve Stored Procedure Uygulamaları

Örnek 17

Satis Tablosunda olup, Urun tablosunda bulunmayan ürünleri veren sql kodunu yazınız.

```
-- Satıs tablosundaki tüm ürün
-- numaralarına erişmek için kürsör
-- tanımlayalım.
DECLARE SatisKursor CURSOR  FOR SELECT
UrunNo FROM Satis

-- Kürsörde okuduğumuz değeri değikene
-- Aktarmak için değişken tanımı
DECLARE @UrunNo char(10)

-- Kürsörü aç
OPEN SatisKursor

-- İlk pozisyona git ve bulunulan
-- Pozisyondaki değeri @UrunNo ya
-- aktar
FETCH NEXT FROM SatisKursor INTO @UrunNo

-- Her satıra tek tek ulaş
WHILE @@FETCH_STATUS=0
BEGIN

-- Kürsörde okunan @UrunNo ürün
-- tablosunda var mı?
IF NOT EXISTS (SELECT * FROM Urun WHERE
UrunNo = @UrunNo)
  PRINT @UrunNo + ' Kodlu Ürün Urun
                          Tablosunda Yok'
FETCH NEXT FROM SatisKursor INTO @UrunNo
END

CLOSE SatisKursor
DEALLOCATE SatisKursor
GO
```

Örnek 18

17. örnektekini spUrunVarmi isimli bir prosedure olarak kaydediniz.

```
CREATE PROCEDURE spUrunVarmi
AS
DECLARE SatisKursor CURSOR  FOR SELECT
UrunNo FROM Satis

DECLARE @UrunNo char(10)

OPEN SatisKursor

FETCH NEXT FROM SatisKursor INTO @UrunNo

WHILE @@FETCH_STATUS = 0
BEGIN
IF NOT EXISTS (SELECT * FROM Urun WHERE
UrunNo = @UrunNo)
PRINT @UrunNo + ' Kodlu Ürün Urun
Tablosunda Yok'
FETCH NEXT FROM SatisKursor INTO @UrunNo
END

CLOSE SatisKursor
DEALLOCATE SatisKursor
GO
```

Örnek 19

18. Soruda oluşan prosedürü çalıştırınız

```
EXEC spUrunVarmi
```

Örnek 20

Müşteri tablosunda ki müşterilerden faturası olanları veren sorguyu yazınız.

```
-- Müşteri tablosundaki tüm müşteri
-- numaralarına erişmek için kürsör
-- tanımlayalım.
DECLARE MusteriKursor CURSOR  FOR SELECT
mno FROM Musteri

DECLARE @mno char(10)

OPEN MusteriKursor

FETCH NEXT FROM MusteriKursor INTO @mno

WHILE @@FETCH_STATUS=0
BEGIN

-- Kürsörde okunan @mno ürün
-- tablosunda var mı?
IF EXISTS (SELECT * FROM Fatura WHERE
mno = @mno)
  PRINT str(@mno) + ' Nolu müşterinin
Faturası Var'
FETCH NEXT FROM MusteriKursor INTO @mno
END

CLOSE MusteriKursor
DEALLOCATE MusteriKursor
GO
```